Math 156: Problem Set 1

Owen Jones

April 9, 2024

1. Given $D = \{x_1, x_2, \ldots, x_n\}$ is i.i.d, $p(D|\mathbf{w}) = \prod_{i=1}^{n} p(x_i|\mathbf{w})$.

$$\lambda_{ML} = \underset{\lambda}{\operatorname{argmax}}\, p(D|\lambda) = \underset{\lambda}{\operatorname{argmax}} \prod_{i=1}^{n} p(x_i|\lambda) = \underset{\lambda}{\operatorname{argmax}} \prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}$$
$$= \underset{\lambda}{\operatorname{argmax}} \log(\prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}) = \underset{\lambda}{\operatorname{argmax}} \sum_{i=1}^{n} \log(\frac{\lambda^{x_i}}{x_i!} e^{-\lambda})$$
$$= \underset{\lambda}{\operatorname{argmax}} \sum_{i=1}^{n} x_i \log(\lambda) - \log(x_i!) - \lambda$$
$$= \underset{\lambda}{\operatorname{argmax}} -n\lambda + \log(\lambda) \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \log(x_i!)$$
$$\Leftrightarrow \frac{d \log(p(D|\lambda))}{d\lambda} = -n + \frac{1}{\lambda} \sum_{i=1}^{n} x_i = 0$$
$$\Rightarrow n = \frac{1}{\lambda} \sum_{i=1}^{n} x_i \Rightarrow \lambda_{ML} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

2.

$$\mathbf{w}_{ML} = \underset{\mathbf{w}}{\text{argmax}}\, p(\mathbf{t}|\mathbf{x}, \mathbf{x}, \beta) = \underset{\mathbf{w}}{\text{argmax}} \prod_{n=1}^{N} \mathcal{N}(t_n|y(x_n, \mathbf{w}), \beta^{-1})$$

$$= \underset{\mathbf{w}}{\text{argmax}} \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} e^{-\frac{\beta}{2}(t_n - y(x_n,\mathbf{w}))^2}$$

$$= \underset{\mathbf{w}}{\text{argmax}}\, \log(\prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} e^{-\frac{\beta}{2}(t_n - y(x_n,\mathbf{w}))^2})$$

$$= \underset{\mathbf{w}}{\text{argmax}}\, \frac{n}{2}\log(\frac{\beta}{2\pi}) - \frac{\beta}{2}\sum_{n=1}^{N}(t_n - y(x_n, \mathbf{w}))^2$$

the constant $\frac{n}{2}\log(\frac{\beta}{2\pi})$ doesn't change minimizer $\mathbf{w}_{ML}$

argmax $f(x) =$ argmin $-f(x)$ thus:

$$\Leftrightarrow \underset{\mathbf{w}}{\text{argmin}}\, E(\mathbf{w}) = \underset{\mathbf{w}}{\text{argmin}}\, \frac{\beta}{2}\sum_{n=1}^{N}(t_n - y(x_n, \mathbf{w}))^2$$

$$\Leftrightarrow \nabla E(\mathbf{w}) = 0 \Rightarrow \frac{\partial E(\mathbf{w})}{\partial i} = 0 \text{ by FONC}$$

By Chain Rule

$$\frac{\partial E(\mathbf{w})}{\partial w_i} = \beta\sum_{n=1}^{N}(t_n - y(x_n, \mathbf{w}))x_n^i = \beta\sum_{n=1}^{N}(t_n - \sum_{j=0}^{M} w_j x_n^j)x_n^i = 0$$

$$\Rightarrow \sum_{j=0}^{M} w_j \underbrace{\sum_{n=1}^{N} x_n^{i+j}}_{A_{ij}} = \underbrace{\sum_{n=1}^{N} t_n x_n^i}_{T_i}$$

Hence, $\mathbf{w}_{ML}$ solves the set of linear equations $\sum_{j=0}^{M} A_{ij} w_j = T_i$

3. Below is a pdf of my code including the function used to generate the curve of best fit and a graph with a scatterplot of the imported data and the curve of best fit for the chosen number of coefficients. As M increases, the upper bound for the relative error between the exact and numerical solution for w grows exponentially. Chose $M = 8$ because there is a large decrease in the least square error between $M = 7$ and $M = 8$, and while the upper bound for the relative error is much larger than for $M = 7$, it is significantly smaller than for any $M > 8$.

In [1]:
```
import pandas as pd
import numpy as np
from numpy import linalg
from matplotlib import pyplot as plt
```

In [2]:
```
column_names=['x_n','t_n']
df=pd.read_csv('hw1-fitting.csv',index_col=0,names=column_names,header=None)
```

In [3]:
```
df
```

Out[3]:

	x_n	t_n
1	0.000000	0.991459
2	0.105263	0.360328
3	0.210526	0.558448
4	0.315789	0.265560
5	0.421053	-1.364200
6	0.526316	-1.983883
7	0.631579	-1.551820
8	0.736842	-0.020161
9	0.842105	1.164831
10	0.947368	1.090539
11	1.052632	1.925967
12	1.157895	1.031809
13	1.263158	0.099923
14	1.368421	0.608555
15	1.473684	-0.701440
16	1.578947	0.566558
17	1.684211	1.998774
18	1.789474	1.423031
19	1.894737	2.386509
20	2.000000	3.199598

In [12]:
```
def polynomial_fit(M,df,plot=True):
    x_n=np.array(df.iloc[:,0])
    t_n=np.array(df.iloc[:,1])
    X, Y = np.meshgrid(np.arange(M+1),np.arange(M+1))
    Z=X+Y
    A=np.array([[sum(x_n**el) for el in row] for row in Z])
    T=np.array([(x_n**i).dot(t_n) for i in np.arange(M+1)])
    w=linalg.solve(A,T)
```

```
    y=lambda x,w:(x**np.arange(M+1)).dot(w)
    y_out=np.array([y(x,w) for x in x_n])
    if plot==True:
        plt.scatter(data=df,x='x_n',y='t_n')
        plt.plot(x_n,y_out)
    return sum((t_n-y_out)**2),linalg.cond(A)*(linalg.norm(A.dot(w)-T))/linalg.r
```

In [13]:

```
np.array([polynomial_fit(m,df,plot=False) for m in np.arange(20)])
```

Out[13]:

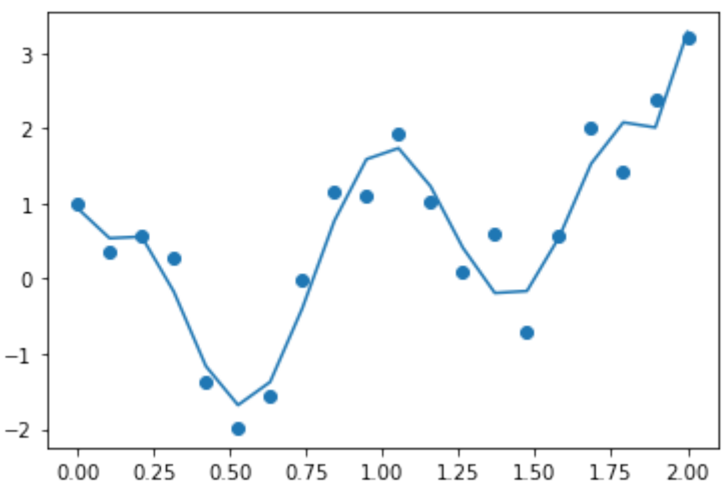
```
array([[3.29063913e+01, 0.00000000e+00],
       [2.31810168e+01, 0.00000000e+00],
       [1.80789209e+01, 1.27218674e-13],
       [1.78928603e+01, 6.54113047e-12],
       [1.25905892e+01, 1.09983879e-09],
       [1.24301870e+01, 2.82526058e-08],
       [6.53579236e+00, 9.18376756e-05],
       [6.41178331e+00, 1.65948250e-02],
       [2.83373721e+00, 4.59882926e+00],
       [2.41764455e+00, 2.45487588e+03],
       [2.40936261e+00, 5.74473863e+04],
       [2.39222170e+00, 9.96031035e+07],
       [2.80859989e+00, 1.52624673e+10],
       [1.47974697e+00, 5.30688853e+09],
       [1.49640674e+00, 1.51010745e+11],
       [1.57784583e+00, 4.26614077e+11],
       [1.41350181e+00, 1.15542284e+12],
       [1.39459659e+00, 2.13670827e+12],
       [1.35016384e+00, 5.11011737e+13],
       [1.34495099e+00, 1.57864938e+13]])
```

In [15]:

```
polynomial_fit(8,df)
```

Out[15]:

```
(2.8337372149712285, 4.598829261330154)
```



In [ ]: